

**IN THE CLAIMS**

The following claim set replaces all prior versions, and listings, of claims in the application:

1. (original) A data processing apparatus, comprising:
  - a main processor operable to execute a sequence of instructions, the main processor comprising a first pipeline having a first plurality of pipeline stages;
  - a coprocessor operable to execute coprocessor instructions in said sequence of instructions, the coprocessor comprising a second pipeline having a second plurality of pipeline stages, and each coprocessor instruction being arranged to be routed through both the first pipeline and the second pipeline; and
  - at least one synchronising queue coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines, the predetermined pipeline stage being operable to cause a token to be placed in the synchronising queue when processing a coprocessor instruction and the partner pipeline stage being operable to process that coprocessor instruction upon receipt of the token from the synchronising queue, thereby synchronising the first and second pipelines between the predetermined pipeline stage and the partner pipeline stage.

2. (original) A data processing apparatus as claimed in Claim 1, further comprising a plurality of said synchronising queues, each said synchronising queue

coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines.

3. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is an instruction queue, the predetermined pipeline stage is in the first pipeline and is arranged to cause a token identifying a coprocessor instruction to be placed in the instruction queue, and the partner pipeline stage is in the second pipeline and is operable upon receipt of the token to begin processing the coprocessor instruction identified by the token.

4. (original) A data processing apparatus as claimed in Claim 3, wherein the predetermined pipeline stage is a fetch stage in the first pipeline and the partner pipeline stage is a decode stage in the second pipeline, that decode stage being operable to decode the coprocessor instruction upon receipt of the token.

5. (original) A data processing apparatus as claimed in Claim 4, wherein the fetch stage in the first pipeline is operable to cause a token to be placed in the instruction queue for each instruction in the sequence of instructions, and the decode stage in the second pipeline is arranged to decode each instruction upon receipt of the associated token in order to determine whether that instruction is a coprocessor instruction that requires further processing by the coprocessor.

6. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is a cancel queue, the predetermined pipeline stage is in the first pipeline and is arranged to cause to be placed in the cancel queue a token identifying whether a coprocessor instruction at that predetermined pipeline stage is to be cancelled, and the partner pipeline stage is in the second pipeline and is operable upon receipt of the token from the cancel queue, and if the token identifies that the coprocessor instruction is to be cancelled, to cause that coprocessor instruction to be cancelled.

7. (original) A data processing apparatus as claimed in Claim 6, wherein the predetermined pipeline stage is an issue stage in the first pipeline, and the partner pipeline stage is a stage following an issue stage in the second pipeline.

8. (original) A data processing apparatus as claimed in Claim 6, wherein the partner pipeline stage is operable upon receipt of the token from the cancel queue, and if the token identifies that the coprocessor instruction is to be cancelled, to remove the coprocessor instruction from the second pipeline.

9. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is a finish queue, the predetermined pipeline stage is in the first pipeline and is arranged to cause to be placed in the finish queue a token

identifying permission for a coprocessor instruction at that predetermined pipeline stage to be retired from the second pipeline, and the partner pipeline stage is in the second pipeline and is operable upon receipt of the token from the finish queue, and if the token identifies that the coprocessor instruction is permitted to be retired, to cause that coprocessor instruction to be retired.

10. (original) A data processing apparatus as claimed in Claim 9, wherein the predetermined pipeline stage is a write back stage in the first pipeline, and the partner pipeline stage is a write back stage in the second pipeline.

11. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is a length queue, the predetermined pipeline stage is in the second pipeline and is arranged, for a vectored coprocessor instruction, to cause to be placed in the length queue a token identifying length information for the vectored coprocessor instruction, and the partner pipeline stage is in the first pipeline and is operable upon receipt of the token from the length queue to factor the length information into the further processing of the vectored coprocessor instruction within the first pipeline.

12. (original) A data processing apparatus as claimed in Claim 11, wherein the predetermined pipeline stage is a decode stage in the second pipeline, and the partner pipeline stage is a first execute stage in the first pipeline.

13. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is an accept queue, the predetermined pipeline stage is in the second pipeline and is arranged to cause to be placed in the accept queue a token identifying whether a coprocessor instruction in that predetermined pipeline stage is to be accepted for execution by the coprocessor, and the partner pipeline stage is in the first pipeline and is operable upon receipt of the token from the accept queue, and if the token identifies that the coprocessor instruction is not to be accepted, to cause that coprocessor instruction to be rejected by the main processor.

14. (original) A data processing apparatus as claimed in Claim 13, wherein the predetermined pipeline stage is an issue stage in the second pipeline, and the partner pipeline stage is a second execute stage in the first pipeline.

15. (original) A data processing apparatus as claimed in Claim 14, wherein the partner pipeline stage is operable upon receipt of the token from the accept queue, and if the token identifies that the coprocessor instruction is not to be accepted, to remove the coprocessor instruction from the first pipeline.

16. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is a store queue used when the coprocessor instruction is a store instruction operable to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the predetermined pipeline stage is in the second pipeline and is arranged, when processing one of said store instructions, to cause to be placed in the store queue a token identifying each data item to be transferred, and the partner pipeline stage is in the first pipeline and is operable upon receipt of each token from the store queue, to cause the corresponding data item to be transferred to the memory.

17. (original) A data processing apparatus as claimed in Claim 16, wherein the predetermined pipeline stage is an issue stage in the second pipeline, and the partner pipeline stage is an address generation stage in the first pipeline.

18. (original) A data processing apparatus as claimed in Claim 1, wherein one of the at least one synchronising queues is a load queue used when the coprocessor instruction is a load instruction operable to cause data items to be transferred from memory accessible by the main processor to the coprocessor, the predetermined pipeline stage is in the first pipeline and is arranged, when processing one of said load instructions, to cause to be placed in the load queue a token identifying each data item to

be transferred, and the partner pipeline stage is in the second pipeline and is operable upon receipt of each token from the load queue, to cause the corresponding data item to be transferred to the coprocessor.

19. (original) A data processing apparatus as claimed in Claim 17, wherein the predetermined pipeline stage is a write back stage in the first pipeline, and the partner pipeline stage is a write back stage in the second pipeline.

20. (original) A data processing apparatus as claimed in Claim 18 wherein one of the at least one synchronising queues is a store queue used when the coprocessor instruction is a store instruction operable to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the predetermined pipeline stage is in the second pipeline and is arranged, when processing one of said store instructions, to cause to be placed in the store queue a token identifying each data item to be transferred, and the partner pipeline stage is in the first pipeline and is operable upon receipt of each token from the store queue, to cause the corresponding data item to be transferred to the memory, and wherein the load instruction and store instruction may be vectored coprocessor instructions defining multiple data items to be transferred, and the apparatus further comprises flow control logic, associated with at least one of the load queue and the store queue, operable to send a control signal to the predetermined pipeline

stage to stop issuance of tokens by the predetermined pipeline stage whilst it is determined that the associated load or store queue may become full.

21. (original) A data processing apparatus as claimed in Claim 20, wherein the flow control logic is provided for the store queue, the flow control logic being operable to issue the control signal upon receiving an indication from the main processor that the partner pipeline stage cannot accept a data item.

22. (original) A data processing apparatus as claimed in Claim 21, wherein the load queue is a double buffer.

23. (original) A data processing apparatus as claimed in Claim 1, wherein each token includes a tag which identifies the coprocessor instruction to which the token relates.

24. (original) A data processing apparatus as claimed in Claim 23, wherein the main processor is operable, when it is necessary to flush coprocessor instructions from both the first and the second pipeline, to broadcast a flush signal to the coprocessor identifying the tag relating to the oldest instruction that needs to be flushed, the coprocessor being operable to identify that oldest instruction from the tag and to flush

from the second pipeline that oldest instruction and any later instructions within the coprocessor.

25. (original) A data processing apparatus as claimed in Claim 24, wherein one or more of said at least one synchronising queues are flushed in response to said flush signal, with the tag being used to identify which tokens within the queue are to be flushed.

26. (original) A data processing apparatus as claimed in Claim 1, wherein the at least one synchronising queue comprises a First-In-First-Out (FIFO) buffer having a predetermined number of entries for storing tokens.

27. (original) A data processing apparatus as claimed in Claim 1, wherein a plurality of said coprocessors are provided, with each synchronising queue coupling a pipeline stage in the main processor with a pipeline stage in one of the coprocessors.

28. (original) A data processing apparatus as claimed in Claim 1, wherein the data processing apparatus has a synchronous design, such that the tokens are caused to be placed in the queue by the predetermined pipeline stage and are caused to be received from the queue by the partner pipeline stage upon changing edges of a clock cycle.

29. (original) A method of synchronisation between pipelines in a data processing apparatus, the data processing apparatus comprising a main processor operable to execute a sequence of instructions and a coprocessor operable to execute coprocessor instructions in said sequence of instructions, the main processor comprising a first pipeline having a first plurality of pipeline stages, and the coprocessor comprising a second pipeline having a second plurality of pipeline stages, and each coprocessor instruction being arranged to be routed through both the first pipeline and the second pipeline, the method comprising the steps of:

- (a) coupling a predetermined pipeline stage in one of the pipelines with a partner pipeline stage in the other of the pipelines via a synchronising queue;
- (b) placing a token in the synchronising queue when the predetermined pipeline stage is processing a coprocessor instruction;
- (c) upon receipt of the token from the synchronising queue by the partner pipeline stage, processing the coprocessor instruction within the partner pipeline stage; whereby synchronisation of the first and second pipelines between the predetermined pipeline stage and the partner pipeline stage is obtained.

30. (original) A method as claimed in Claim 29, wherein a plurality of said synchronising queues are provided, and said steps (a) to (c) are performed for each synchronising queue.

31. (original) A method as claimed in Claim 29, wherein one of the at least one synchronising queues is an instruction queue, the predetermined pipeline stage is in the first pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), placing a token in the instruction queue identifying a coprocessor instruction; and

at said step (c), upon receipt of the token, beginning processing of the coprocessor instruction identified by the token within the partner pipeline stage.

32. (original) A method as claimed in Claim 29, wherein one of the at least one synchronising queues is a cancel queue, the predetermined pipeline stage is in the first pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), placing a token in the cancel queue identifying whether a coprocessor instruction at that predetermined pipeline stage is to be cancelled; and

at said step (c), upon receipt of the token from the cancel queue by the partner pipeline stage, and if the token identifies that the coprocessor instruction is to be cancelled, causing that coprocessor instruction to be cancelled.

33. (original) A method as claimed in Claim 29, wherein one of the at least one synchronising queues is a finish queue, the predetermined pipeline stage is in the first

pipeline and the partner pipeline stage is in the second pipeline, the method comprising the steps of:

at said step (b), placing in the finish queue a token identifying permission for a coprocessor instruction at that predetermined pipeline stage to be retired from the second pipeline; and

at said step (c), upon receipt of the token from the finish queue by the partner pipeline stage, and if the token identifies that the coprocessor instruction is permitted to be retired, causing that coprocessor instruction to be retired.

34. (original) A method as claimed in Claim 29, wherein one of the at least one synchronising queues is a length queue, the predetermined pipeline stage is in the second pipeline and the partner pipeline stage is in the first pipeline, and the method comprises the steps of:

at said step (b), for a vectored coprocessor instruction, placing in the length queue a token identifying length information for the vectored coprocessor instruction; and

at said step (c), upon receipt of the token from the length queue by the partner pipeline stage, factoring the length information into the further processing of the vectored coprocessor instruction within the first pipeline.

35. (original) A method as claimed in Claim 29, wherein one of the at least one synchronising queues is an accept queue, the predetermined pipeline stage is in the

second pipeline and the partner pipeline stage is in the first pipeline, the method comprising the steps of:

at said step (b), placing in the accept queue a token identifying whether a coprocessor instruction in that predetermined pipeline stage is to be accepted for execution by the coprocessor; and

at said step (c), upon receipt of the token from the accept queue by the partner pipeline stage, and if the token identifies that the coprocessor instruction is not to be accepted, causing that coprocessor instruction to be rejected by the main processor.

36. (original) A method as claimed Claim 29, wherein one of the at least one synchronising queues is a store queue used when the coprocessor instruction is a store instruction operable to cause data items to be transferred from the coprocessor to memory accessible by the main processor, the predetermined pipeline stage is in the second pipeline and the partner pipeline stage is in the first pipeline, the method comprising the steps of:

at said step (b), when processing one of said store instructions, placing in the store queue a token identifying each data item to be transferred; and

at said step (c), upon receipt of each token from the store queue by the partner pipeline stage, causing the corresponding data item to be transferred to the memory.

37. (currently amended) A method as claimed in ~~any of claims 29 to 36~~,  
wherein one of the at least one synchronising queues is a load queue used when the  
coprocessor instruction is a load instruction operable to cause data items to be transferred  
from memory accessible by the main processor to the coprocessor, the predetermined  
pipeline stage is in the first pipeline and the partner pipeline stage is in the second  
pipeline, the method comprising the steps of:

at said step (b), when processing one of said load instructions, placing in the load  
queue a token identifying each data item to be transferred; and  
at said step (c), upon receipt of each token from the load queue by the partner  
pipeline stage, causing the corresponding data item to be transferred to the coprocessor.

38. (original) A method as claimed in Claim 37 wherein one of the at least one  
synchronising queues is a store queue used when the coprocessor instruction is a store  
instruction operable to cause data items to be transferred from the coprocessor to memory  
accessible by the main processor, the predetermined pipeline stage is in the second  
pipeline and the partner pipeline stage is in the first pipeline, the method comprising the  
steps of:

at said step (b), when processing one of said store instructions, placing in the store  
queue a token identifying each data item to be transferred; and  
at said step (c), upon receipt of each token from the store queue by the partner  
pipeline stage, causing the corresponding data item to be transferred to the memory; and

wherein the load instruction and store instruction may be vectored coprocessor instructions defining multiple data items to be transferred, and the method further comprises the step of:

(d) for at least one of the load queue and the store queue, sending a control signal to the predetermined pipeline stage to stop issuance of tokens by the predetermined pipeline stage whilst it is determined that the associated load or store queue may become full.

39. (original) A method as claimed in Claim 38, wherein said step (d) is performed for the store queue, at said step (d) the method comprising the step of issuing the control signal upon receiving an indication from the main processor that the partner pipeline stage cannot accept a data item.

40. (original) A method as claimed in Claim 29, wherein each token includes a tag which identifies the coprocessor instruction to which the token relates.

41. (original) A method as claimed in Claim 40, wherein, when it is necessary to flush coprocessor instructions from both the first and the second pipeline, the method further comprises the steps of:

broadcasting a flush signal from the main processor to the coprocessor identifying the tag relating to the oldest instruction that needs to be flushed;

within the coprocessor, identifying from the tag that oldest instruction and flushing from the second pipeline that oldest instruction and any later instructions within the coprocessor.

42. (original) A method as claimed in Claim 41, further comprising the step of flushing one or more of said at least one synchronising queues in response to said flush signal, with the tag being used to identify which tokens within the queue are to be flushed.

43. (original) A method as claimed in Claim 29, wherein the at least one synchronising queue comprises a First-In-First-Out (FIFO) buffer having a predetermined number of entries for storing tokens.

44. (original) A method as claimed in Claim 29, wherein a plurality of said coprocessors are provided, with each synchronising queue coupling a pipeline stage in the main processor with a pipeline stage in one of the coprocessors.

45. (original) A method as claimed in Claim 29, wherein the data processing apparatus has a synchronous design, such that the tokens are placed in the queue by the predetermined pipeline stage and are received from the queue by the partner pipeline stage upon changing edges of a clock cycle.